IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants	: Wolfgang Theilmann, et al.	Art Unit	: 2166
Serial No.	: 10/809,873	Examiner	: Navneet K. Ahluwalia
Filed	: March 25, 2004	Confirmation No.: 7587	
Title	: VERSIONING ELECTRONIC LEARNING OBJECTS USING PROJECT OBJECTS		

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

PRE-APPEAL BRIEF REQUEST FOR REVIEW

We request that a panel of Examiners review the rejections made by the Examiner because of the deficiencies discussed below.

I. Rejections

Claims 14 to 26 and 28 were rejected under §101. All of the claims were rejected under §103 over U.S. Patent Publication No. 2002/0178181 (Subramanyan) in view of U.S. Patent Publication No. 2004/0002049 (Beavers).

II. Questions For Review

We respectfully request the panel to review the following issues: (1) whether claims 14 to 26 and 28 were properly rejected under §101; and (2) whether the claims were properly rejected over Subramanyan in view of Beavers. We reserve the right to expand the issues or to present new issues when filing an appeal brief.

III. Independent Claim 14

Independent claim 14 recites a computer program product "embodied in a tangible computer-readable medium". This language excludes unpatentable matter, such as propagated signals. Withdrawal of the §101 rejection of claims 14 to 26 and 28 is respectfully requested.

IV. Independent Claims 1 and 14

Independent claim 1 recites

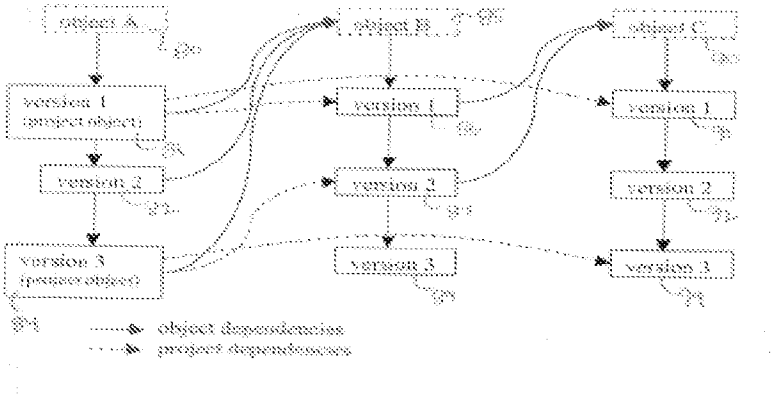
> 1. A method, performed by one or more processing devices, for use in an electronic learning system that stores information as learning objects, the method comprising:
> designating a target learning object as a project object; and

storing version dependency data in the project object, the version dependency data identifying versions of other learning objects upon which the project object depends, the other learning objects including at least a version of a first object upon which the project object directly depends, and a version of a second object upon which the project object indirectly depends, the project object being an object that is separate from the first object and the second object;

wherein the other learning objects, including versions of the first and second objects, do not store version dependency data, and wherein the other learning objects store dependency data that identifies an object dependency but that does not identify a version dependency, the other learning objects relying on version dependency data in the project object for identification of version dependency.

As set forth in the underlined portion above, claim 1 recites different types of dependency data, namely "**dependency data**" and "**version dependency data**", the combination of which, as claimed, is not found in the applied references.

An embodiment, which illustrates the invention of claim 1, is shown in Fig. 8 of the application, which is reproduced below.



As explained in our previous responses, a project object 81 stores version dependency data, which is identified by the dashed lines. The version dependency data identifies the versions of other objects, here object 85 and object 90, upon which the project object 81 depends. In this example, object 85 exists in three versions: 86, 87 and 89; and object 90 exists in three versions: 91, 92, and 94. The version dependency data identifies the version of the object upon which the project object depends. A first object, e.g., object 85, contains dependency data, which is identified by solid lines. The dependency data identifies a second object, e.g., object 90, upon which the first object 85 depends, but does not identify the version upon which object 90 depends. This version dependency data is stored in the project object. The other learning objects therefore rely on the project object for identification of version dependency.

Subramanyan was cited, on pages 4 and 5 of the Office Action, for its alleged disclosure of the features of claim 1, except for "the object and version dependency data", which Beavers was alleged to disclose. In the "response to arguments" section on pages 2 and 3 of the Office Action, the Examiner alleges the following:

> *In response to Applicant's argument, the Examiner submits that Subramanyan in combination with Beavers teaches the version dependency data identifying versions of other learning objects and dependency data identifying object dependency. This is clearly disclosed in Subramanyan as it teaches versions in paragraph 16 where it talks about there being version control tools and storing the versions and it can be worked on to modify and also to revert back to previous version. Furthermore, Subramanyan in paragraphs 31 and 33 teach the learning object via a version control tool and an authoring tool equating to the version dependency data and also the templates and the subject matter groups showing the dependency data. Furthermore, Beavers teaches the object and version dependency data specifically in detail in* paragraphs 149 and 174.

We disagree with these statements in the Office Action. More specifically, Subramanyan says very little about its version control tool, and certainly nothing that would imply that Subramanyan's versions/version control tool have/has the structure ascribed to it by the Office Action. Subramanyan's version control tool is only mentioned in the following excerpts taken from the following paragraphs of Subramanyan:

> [0016] … a built-in version control that enables team members and users to view changes in their work and revert back to older versions.
>
> [0031] The storyboard server comprises a number of enabling software tools and databases, including a storage database…a version control tool...
>
> [0035] Each of these tools comprising the storyboard server… work in conjunction with other storyboard tools to facilitate and integrate the storyboarding process, including a messaging tool, a version control tool….
>
> [0036] The version control tool enables users to access, edit, copy, etc., previous versions of work contained in the storyboard server, including templates, empty learning objects and learning paths, frames, pages, and edited content or subject matter, as well as any textual information.
>
> [0040] … The storyboard also comprises messaging and tracking or version control.
>
> [0048] … As templates, learning objects, and frames are modified, a history of these items is stored in the storyboard server. Users can revert back to previous versions….

> [0056] …At one point, the SME wants to go back to an earlier version of a screen where the new changes didn't quite work right. The Storyboard Server's version control makes this possible.

Except for the claims, the entire description of Subramanyan's version control tool is set forth above in bold and underlining. As you can see, Subramanyan's version control involves storing a history of on-screen items, and reverting to that history, if necessary. There is no description whatsoever in Subramanyan of storing version dependency data or dependency data in an object, much less a project object, as claimed. There is also no description in Subramanyan of using separate version dependency data or dependency data to identify dependencies, as claimed. For example, there is no description in Subramanyan of using data in a project object for identification of version dependency of another object.

Paragraphs 31 and 33, which were referenced in the "response to argument section" merely describe creation of learning objects. These paragraphs are not understood to disclose versions of the type claimed. Neither are the following paragraphs, which were cited from the summary section of Subramanyan: 9, 12, 16 and 19 to 21.

On page 5, the Office Action further states:

> Subramanyan does not explicitly disclose the object and version dependency as claimed.
>
> Beavers, however teaches the object and the version dependency as claimed in paragraphs 149 and 174.

We disagree with this characterization of Beavers. Relevant portions of the cited paragraphs of Beavers are reproduced below:

> [0149] … In tested versions of the learning system, the forward error correction took the form of a simple process of broadcasting multiple versions of the presentation data in case a packet of any one of the repeated broadcast is lost or corrupted during transit over the network. The aforementioned data entry field is used to specify how many times a packet of the presentation data is to be sent. In tested versions of the system this could be varied between 1 and 5 times, with 2 being the default.
>
> [0174] … In tested versions of the present system, this list included only one listing--namely an "About" link. Selecting the about link causes an information box to appear in the slideshow window. This information box is a standard "about" box and includes information such as the title of the program and its version number…

Respectfully, we do not at all see a correlation between the cited paragraphs of Beavers and what is claimed. Regarding the dependency data and the version dependency data, claim 1 recites the following:

the version dependency data identifying versions of other learning objects upon which the project object depends

the other learning objects store dependency data that identifies an object dependency but that does not identify a version dependency, the other learning objects relying on version dependency data in the project object for identification of version dependency.

These features are not found in the cited paragraphs of Beavers, or elsewhere in Beavers for that matter. As we understand it, Beavers includes different system versions, including tested versions. Beavers also describes "broadcasting multiple versions of the presentation data in case a packet of any one of the repeated broadcast is lost or corrupted during transit over the network". We do not understand how this information could correlate to the dependency data and the version dependency data that is claimed. For example, the version dependency data, as claimed, identifies versions of other learning objects upon which the project object depends. None of the Beavers data is indicated to identify object dependencies. The dependency data, as claimed, identifies an object dependency but that does not identify a version dependency. There is no indication that any version data in Beavers (to the extent that there is any) identifies an object dependency but not a version dependency.

Given what we believe to be the glaring deficiencies of both Subramanyan and Beavers, even if the two references were combined, the resulting hypothetical combination would fail to render claim 1 obvious. Accordingly, claim 1 is believed to be patentable. Independent claim 14 is believed to be patentable for at least the same reasons explained above.

Please apply any deficiency in fees to deposit account 06-1050, referencing the attorney docket number 13909-161001.

Respectfully submitted,

Date: January 28, 2009

/Paul Pysher/
Paul A. Pysher
Reg. No. 40,780

Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110
Telephone: (617) 542-5070
Facsimile: (617) 542-8906